The Design of an Image Bank for Astronomical Images

Robert J Lucas
Department of Physics and Astronomy
The Open University

Abstract: Image Banks, which are collections of images with associated data and captions, are a valuable teaching tool for Astronomy courses at the Open University. Until now web pages have been created for each image and its associated information. This paper examines how a database, front-ended by a multimedia authoring tool, can provide a much more flexible and maintainable architecture for producing Image Banks. Accessibility issues are discussed.

Introduction

Image Banks are used within several Astronomy courses presented by the Open University. They typically consist of between one and two hundred images with titles, captions (often running to a thousand words or more), dates, credits and scale information. The idea is that these images can be browsed by logical sections or by searches on the captions. The student might be investigating a certain kind of object. Sometimes exercises can be carried out on the images such as measurement of a particular feature. The image banks are perceived as being vital components of the relevant courses.

Until recently these image banks have been created by using Dream-Weaver to create a set of individual web-pages that incorporate the image and the various text fields. Hence a particular image and its various text fields will become a single web page. The web-page approach has some advantages: everyone has a browser with which they can view the image bank, so there's nothing to install and the image bank will run straight from the CD/DVD. Also screen readers such as JAWS (Job Access With Speech) are able to read the text for partially sighted and blind users.

It is foreseen that Image Banks will be used more widely, thus a more generic approach to creating and maintaining them is seen as highly desirable. Ideally, we would like to have a viewer which can be used with any set of images so that the viewer and the sets of data it can use can be maintained independently of each other.

Requirements for an Image Bank Viewer.

(1) A generic approach can be taken to viewing the images that does not require each image to be individually processed to produce a web-page or some such viewing medium. It shouldn't be necessary to produce pages that list all the hyperlinks to sections and subsections. These should be generated automatically from the data.
(2) The viewer must not rely on the user having any particular application installed unless this is something that the user must have anyway (such as a web browser).
(3) It should not need to be installed.
(4) It should run directly from the CD/DVD.
(5) We want to be able to pan and zoom images.
(6) Measurement tools should be incorporated into the viewer.
(7) Accessibility should be addressed.

Proposing a solution

The first requirement suggests a database approach with each image being associated with a record in a table. This table would store all the textual items and give the path to the particular image. A page can then be created that can display *any* of the images with its associated data fields. Additionally, pages giving hyperlinks to each of the sections can be generated at run-time.

The second requirement rules out using Power Point, Word or whatever as either the application delivery mechanism or as an Active X component for viewing images or documents.

The third requirement makes component-based applications unusable. Hence, VB, C, C++ and Delphi are all going to be problematic as development tools.

Requirement 4 limits us to certain multimedia application generators that are designed with exactly this in mind.

Requirement 5 should be available in any programming environment and we would certainly expect a multimedia-authoring package to be able to provide this.

Requirement 6 can be satisfied just a long as a minimal amount of programming is available to us.

Requirement 7 will be discussed in its own section ‘Accessibility’.

Database Connectivity

It is clear that an Image Bank that is required to work with various collections of images and their associated data should be a database application. With the added requirements of needing to operate directly from CD/DVD with no installation then a multimedia development tool with ODBC (Open DataBase Connectivity) is going to be the only solution. These requirements are met by the Opus Professional Multimedia development environment. This has the additional advantage of permitting programming in Java.

ODBC drivers, which are available as part of the Windows Operating System, give access to a huge range of databases some of which are not actually genuine databases at all, but are just treated as such. This is achieved via SQL (Standard Query Language). This allows us to select fields from tables of information, apply conditions on whatever is selected and impose ordering on retrieved datasets. Whether we have a genuine database system such as Oracle, or a table of data in the form of an Excel spreadsheet is of little consequence as we access both using identical SQL strings. There are questions of efficiency and sharing which commercial database systems address but simply don’t apply here where we have a relatively small, non-shared dataset. A fully-fledged database system such as Oracle would not be feasible here. It couldn’t be provided on a runnable CD/DVD, it would require the user to have Oracle installed, and it’s very expensive amongst many other reasons. A PC based system such as Access would provide all that we want, but again it could not be used on a runnable CD as it would need to create a log file at run time which could not be achieved on a read-only device. Hence, an Excel spreadsheet will serve as the database, with the connection to the application provided by a suitable ODBC driver.

Accessibility

It is certainly not clear that Opus can provide the necessary Accessibility features as its controls do not even support a tab ordering which would allow it to be controlled via the keyboard rather than the mouse. Some experimentation has shown that tab-ordering can be achieved by programming it in for each screen, but it is surprisingly clumsy and elaborate. However, it is an easy matter to use keys for the same functions as on-screen buttons. For example, the PgDn key can be used instead of mouse-clicking on the Page Forward button.

Screen readers such as JAWS need access to the underlying text in order to be able to render it as speech. This is only usually the case for the major Microsoft applications like Word where it has been specially provided for or in HTML documents where the source text is not encoded (surprisingly screen readers never do screen-scraping coupled with Optical Character Recognition, they always require special hooks to be provided that give them access to the underlying text).

In addition to tools such as JAWS which are aimed squarely at the partially sighted or blind user, there are a multitude of Text To Speech (TTS) applications available. A common facility is being able to paste text into them and have it converted to WAV or MP3 format, i.e. the text is rendered into a spoken form and stored as a sound file in one of the Windows standard formats. The latest generation of TTS programs, such as Natural Reader from AT&T is really very good with a tonal quality that is good enough to be used where students would prefer to listen to the captions rather than read them. Thus we can take all the text from our captions and other fields and translate them to sound files that can be played by the multimedia application. Additionally we can make help on navigating and using the application available as both text and speech. This actually works better than using an application like JAWS which tends to deliver an unnecessarily large amount of information about the current window. Whereas, our application restricts itself to exactly what the blind user needs to know.

Formatted text

There is a requirement for some formatting of the text such as subscripts used in chemical formulae and superscripts to indicate numerical powers. These are surprisingly awkward to deal with. A superscript character is not simply an item of a character set. It is such an item plus additional attributes set for this item. So when a character string is read in from a database character field there is no such attribute data and any such formatting simply does not exist.

Unicode gives us the capability of storing wide character sets, but these do not address consistently the need for superscripts and subscripts. Typically some values such as 2 for squared and 3 for cubed will exist but little else. For this reason Unicode is simply a red-herring.

The options that remain are to:

- Mark all cases of special formatting with escape sequences so that the formatting can be applied once the data has been extracted into the application.
- Store the data in a standard format such as RTF (Rich Text Format).

Perhaps surprisingly SQL based databases do not have the capability to store any formatted text directly. There is no formatted text type in the SQL Standard. If the second option is to be used then a binary type is needed for a database field which is to store it. Or store each of the captions in its own formatted file which is then referenced in the database.

Having stored it, as binary (which precludes the use of text files or spreadsheets as our database source, something that we might wish to do if we were to use ODBC) then we need to be able to retrieve the formatted text and then display it in its formatted form.

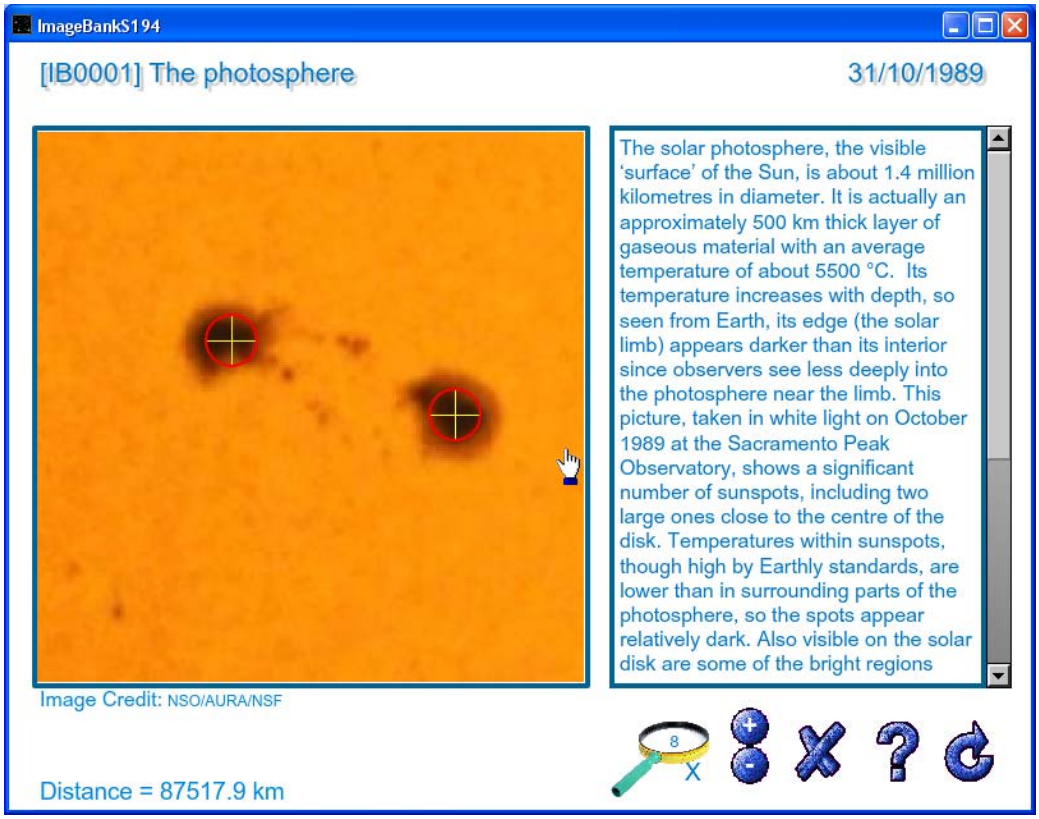
We also need to be mindful that we might wish to change other attributes of the text, such as size for readability.

Opus will allow us to use a document viewer to view any Active X document. However, this requires the application that views this format to be present on the end-user's machine. This clearly cannot be guaranteed in the case of Word or Excel. The only viable format is RTF which can be displayed without recourse to an external application.

Image Bank Design

The design for the Image Bank used an Excel spreadsheet as the database. The Image Bank application was written in Opus and used an ODBC connection to the database. The database stores the captions, the image file paths and other information such as credits and image size. In the first instance, superscripts and subscripts were catered for using escape sequences. This is not ideal and in future we will store the captions in individual RTF files with a reference to them in the caption field of the database.

Features, such as being able to do on-screen measurements, panning and zooming were exceptionally easy to program into the application using the underlying Java language and the available set of multimedia functions.



The Image Bank viewer which shows the measurement of the distance between two sunspots.

Conclusions

A high-level multimedia authoring tool coupled to a database has proved an excellent architecture for the design of the Image Bank. We are now looking at designing a Resource Library from which Image Banks for different courses can be automatically generated from the department's entire collection of image and image-related data. This will require the course editor to simply select the images required for the course causing the entire CD/DVD image to be created. This will remove the hard work currently needed to create a new Image Bank and also remove the possibilities for errors.

Although we haven't as yet incorporated the text to speech technology into our current version (due to licencing difficulties), the effectiveness of this approach is manifestly obvious. Our students can spend their time looking at the images instead of reading the text. Additionally we can provide instructions for using the package as speech generated directly from the Help text.

References:

A Guide to the SQL Standard. C.J. Date with Hug Darwen. Addison Wesley 1993.

Illuminatus Opus Pro User Manual and Additional Features. Digital Workshop Ltd.